

#2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re U.S. Patent Application of)
)
MISAKA et al.)
)
Application Number: To Be Assigned)
)
Filed: Concurrently Herewith)
)
For: REGISTER ALLOCATION METHOD AND SOFTWARE)
DEVELOPMENT METHOD FOR VARIOUS)
EXECUTION ENVIRONMENTS AND LSI FOR)
EXECUTING DEVELOPED SOFTWARE)



Honorable Assistant Commissioner
for Patents
Washington, D.C. 20231

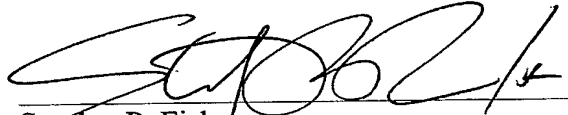
**REQUEST FOR PRIORITY
UNDER 35 U.S.C. § 119
AND THE INTERNATIONAL CONVENTION**

Sir:

In the matter of the above-captioned application for a United States patent, notice is hereby given that the Applicant claims the priority date of December 13, 2000, the filing date of Japanese patent application P2000-378801.

The certified copy of Japanese patent application 2000-378801 is being submitted herewith. Acknowledgment of receipt of the certified copy is respectfully requested in due course.

Respectfully submitted,


Stanley P. Fisher
Registration Number 24,344

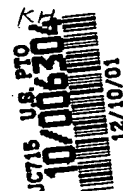
REED SMITH HAZEL & THOMAS LLP
3110 Fairview Park Drive
Suite 1400
Falls Church, Virginia 22042
(703) 641-4200

JUAN CARLOS A. MARQUEZ
Registration No. 34,072

December 10, 2001

31 00-639 05

日 本 国 特 許 庁
JAPAN PATENT OFFICE



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日
Date of Application:

2000年12月13日

CERTIFIED COPY OF
PRIORITY DOCUMENT

出 願 番 号
Application Number:

特願2000-378801

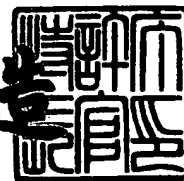
出 願 人
Applicant(s):

株式会社日立製作所

2001年10月19日

特 許 庁 長 官
Commissioner,
Japan Patent Office

及 川 耕 造



【書類名】 特許願

【整理番号】 NT00P0424

【提出日】 平成12年12月13日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 12/02510

【発明者】

 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内

 【氏名】 三坂 智

【発明者】

 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内

 【氏名】 相坂 一夫

【発明者】

 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目280番地 株式会社日立製作所 中央研究所内

 【氏名】 在塚 俊之

【特許出願人】

 【識別番号】 000005108

 【氏名又は名称】 株式会社日立製作所

【代理人】

 【識別番号】 100068504

 【弁理士】

 【氏名又は名称】 小川 勝男

 【電話番号】 03-3661-0071

【選任した代理人】

 【識別番号】 100086656

 【弁理士】

 【氏名又は名称】 田中 恭助

【電話番号】 03-3661-0071

【選任した代理人】

【識別番号】 100094352

【弁理士】

【氏名又は名称】 佐々木 孝

【電話番号】 03-3661-0071

【手数料の表示】

【予納台帳番号】 081423

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 異種実行環境におけるレジスタの割当て方法、異種実行環境におけるソフトウェア開発方法、および、それを実行するプログラムが組み込まれた L S I

【特許請求の範囲】

【請求項 1】 計算機システムの異なった実行環境で実行させる際の C P U 内の汎用レジスタにデータを割当てする方法において、

この計算機システムは、ハードウェアと、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

そのオペレーティングシステム上で実行させる実行プログラムとしては、ユーザ固有の機能からなる実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムに制御を渡すときのパラメータのために用いられるレジスタの数を、

前記ハードウェア内のメモリに引数格納集合部を作り、

前記汎用レジスタにその引数格納集合部をポイントさせることにより、

コンパイラが予約する引数用途格納用レジスタの数を越えない様に調整することを特徴とする異種実行環境におけるレジスタの割当て方法。

【請求項 2】 前記実行環境差異吸収プログラムは、

共通化関数をコンパイルしたモジュールと、それぞれの実行環境に固有のプログラムとからなり、

前記共通化関数の引数として、コンパイラが予約する引数用途格納用レジスタの数を越えない様に調整したソースコードを、コンパイラに入力することを特徴とする請求項 1 記載の異種実行環境におけるレジスタの割当て方法。

【請求項 3】 前記共通化関数の引数として、

前記オペレーティングシステム上で動作させるタスクに関する情報を渡すときに、

その引数として、

前記タスクの制御および管理に用いる識別子を格納するメモリのアドレス値と

前記タスクの実行ルーチンの先頭アドレス値と、
前記タスクが前記ハードウェアを占有する優先度の値と、
前記異種環境を記述するための引数情報集合部を格納する先頭のメモリのアドレス値であることを特徴とする請求項2記載の異種実行環境におけるレジスタの割当て方法。

【請求項4】 前記引数情報集合部が、
前記タスクのタスク属性値と、
前記タスクが、前記ハードウェアを占有する上限時間値と、
前記タスクが、必要とする合計のスタックサイズと、
前記タスク起動時のスタックのアドレスであるスタックベースポインタと、
前記タスクが、前記ハードウェアに装備のマルチメディア命令を使用するか否かの値を設定するCPUモード値と、
前記タスク名称の格納先であるメモリのアドレス値と、
前記タイク起動時に受け渡すためのタスク情報を格納するメモリの先頭アドレス値であることを特徴する請求項3記載の異種実行環境におけるレジスタの割当て方法。

【請求項5】 計算機システムの異なった実行環境で実行させる際の異種実行環境におけるソフトウェア開発方法において、

この計算機システムは、ハードウェアと、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

そのオペレーティングシステム上で実行させる実行プログラムとしては、ユーザ固有の機能からなる実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムに制御を渡すときのパラメータのために用いられるレジスタの数を、

前記ハードウェア内のメモリに引数格納集合部を作り、

前記汎用レジスタにその引数格納集合部をポイントさせることにより、

コンパイラが予約する引数用途格納用レジスタの数を越えない様に調整するよ

うな実行プログラムを生成することを特徴とする異種実行環境におけるソフトウェア開発方法

【請求項6】 前記実行環境差異吸収プログラムは、
共通化関数をコンパイルしたモジュールと、それぞれの実行環境に固有のプログラムとからなり、

前記共通化関数の引数として、コンパイラが予約する引数用途格納用レジスタの数を越えない様に調整したソースコードを、コンパイラに入力することを特徴とする請求項5記載の異種実行環境におけるソフトウェア開発方法。

【請求項7】 異なった実行環境で動作させるLSIにおいて、
このLSIは、ハードウェアと、そのハードウェア上で実行させるオペレーティングシステムとを持ち、

前記ハードウェアとしては、
汎用レジスタを含むCPUと、
メモリと、

専用機能からなるデバイスとからなり、

前記オペレーティングシステム上で実行させる実行プログラムとしては、ユーザ固有の機能からなる実行プログラムと、異なった実行環境の差異を吸収する実行環境差異吸収プログラムとからなり、

前記実行環境差異吸収プログラムに制御を渡すときのパラメータのために用いられるレジスタの数を、

前記メモリに引数格納集合部を作り、

前記汎用レジスタにその引数格納集合部をポイントさせることにより、

コンパイラが予約する引数用途格納用レジスタの数を越えない様に調整するような実行プログラムを前記メモリ上に組み込んだことを特徴とする異種実行環境で動作させるLSI。

【請求項8】 前記実行環境差異吸収プログラムは、
共通化関数をコンパイルしたモジュールと、それぞれの実行環境に固有のプログラムとからなり、

前記共通化関数の引数として、コンパイラが予約する引数用途格納用レジスタ

の数を越えない様に調整したソースコードを、コンパイラに入力することを特徴とする請求項7記載の異種実行環境で動作させるLSI。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、異種実行環境におけるレジスタの割当て方法、および、異種実行環境におけるソフトウェア開発方法に係り、CPUを異なった環境で実行させるときに、移植が容易で、しかも、メモリの使用効率が良く、実行速度を低下させることのない異種実行環境におけるレジスタの割当て方法、および、異種実行環境におけるソフトウェア開発方法に関する。

【0002】

【従来の技術】

従来、異なった環境でコンピュータシステムを実行させるために、開発や移植を容易におこなうための技術が提案されている。

【0003】

このような技術の例として、特開平8-63363号公報の「仮想実行環境システム」がある。

【0004】

以下では、図6を用いてこの仮想実行環境システムについて説明する。

【0005】

図6は、特開平8-63363号公報の「仮想実行環境システム」のシステム概要図である。

【0006】

ユーザ（プログラマ）が作成するアプリケーションソフトウェアのソースコードとしては、プログラム本体11と置換情報記述部12がある。

【0007】

プログラム本体は、アプリケーションソフトウェアの目的ごとに異なるソースコードであり、置換情報記述部12は、プログラム本体11中に記述された情報を、仮想実行環境が実現される既存オペレーティング・システム60の機種毎に

対応して適切な情報に置換える部分である。

【0008】

オペレーティング・システム60で実行するためには、これらのプログラム本体11と置換情報記述部12に記述された置換情報を、既存オペレーティング・システム用コンパイラを利用して、実行可能プログラム30にし、それを実行部22が実行する。翻訳部21は、そのための仮想実行環境上で実行可能な実行可能プログラムに翻訳する部分である。

【0009】

その際に、ソースプログラムをC言語で記述するときには、置換情報記述部12は、コンパイラのプリプロセッサとして処理されるマクロ定義を利用することができる。

【0010】

このようにして、アプリケーションソフトウェア10のプログラム本体11を変更する事無く、様々な環境用に作成された実行可能プログラム30を既存オペレーティング・システム60上で動作させることが可能となる。

【0011】

【発明が解決しようとする課題】

上記従来技術は、様々な実行環境を想定し、それらでプログラムの移植や開発を容易にすることを目的としている。

【0012】

しかしながら、上記従来技術は、実行可能プログラムが実行されるときメモリの使用効率や実行速度について考慮されていない。

【0013】

以下、図7ないし図9を用いて従来技術の課題について説明する。

【0014】

図7は、異なった実行環境でソフトウェア開発をおこなうときの概念図である。

【0015】

図8は、計算機システム上で実行プログラムが実行されるとき呼出し関係を

示した図である。

【0016】

図9は、実行環境差異吸収プログラム-Aを呼出すときのレジスタの割当てを示す図である。

【0017】

ここでは、従来技術をシステム上に実装したときの構成をより詳細に説明して、その問題点を明らかにすることにしよう。

【0018】

システムとして、図7の様に計算機システム-A105a、計算機システム-B105bがあり、そこでプログラムを実行する環境を実行環境-A104a、実行環境-B104bとすることにする。実行環境-A104aは、具体的には、ハードウェア-A106aと、ハードウェアを制御しアプリケーションプログラムとハードウェアの仲立ちをするOS-A107aである。

【0019】

アプリケーション本体100は、ユーザが個々のアプリケーションを記述するソースコードであり、システムから機能の提供を受けるときには、共通化関数102を呼出す。ここで、注意することは、アプリケーション本体100を記述するユーザは、共通化関数インタフェース101を意識すればよく、個々の環境によってコーディングを変えなくても良いことである。したがって、アプリケーション本体100は、ソースレベルでの互換性が保たれることになる。

【0020】

共通化関数102、開発用ライブラリ関数として提供され、ユーザには、共通化関数インタフェースが公開されることになる。共通化関数は、この様に環境に依らない統一したインタフェースをユーザに提供するためのものである。

【0021】

そして、ユーザが記述したソースをコンパイル・リンク作業をおこなって、実行形式を作成することになるが、実行環境-A104aで実行させるときには、図7に示される様に、コンパイラ-A108aでコンパイルし、実行環境-B104bで実行させるときには、コンパイラ-B108bでコンパイルする。

【0022】

また、同様に実行環境-A用のリンカA109aにより、実行環境プログラム-A103aをリンクする。実行環境プログラム-A103aは、実行環境-A104aで実行させるためのプログラムであり、このなかで、OS-A107aのシステムコールがコールされることになる。実行環境-B104bについても同様である。

【0023】

ユーザのソースコードであるアプリケーション本体100に対して、コンパイル、リンクがおこなわれ、この例では、実行環境-A104aと実行環境-B104b用に、それぞれ実行プログラム-A110aと、実行プログラム-B110bが生成される。

【0024】

そして、共通化関数102をコンパイルしたモジュールと、実行環境プログラム-A103a、実行環境プログラム-B103bは、コンパイル・リンクの作業を経て、それぞれの固有の実行環境で実行可能な実行環境差異吸収プログラム-A111aと実行環境差異吸収プログラム-B111bになる。

【0025】

次に、図8を用いてこのような実行環境でプログラムを実行させたときの制御の流れについて説明する。どちらでも同じになるので、以降では実行環境-Aを例にとって説明していくことにする。

【0026】

実行環境-A104aで実行プログラム-A110aに制御を移すと、実行プログラム-A110aでは、共通化関数インターフェースにより、共通化関数102を呼出す。共通化関数102が呼出されると、実行環境差異吸収プログラム-A111aに制御が移ることになる。

【0027】

実行環境差異吸収プログラム-A111aは、システムコールにより、OS-A107aの機能と呼出して利用する。また、OS-A107aは、必要ならハードウェア-A106aの機能を利用する。

【0028】

そして、呼出し元に返り、最後に、実行プログラム-A110aに制御が戻ることになる。

【0029】

実行形式にして実行するときには、関数間で受け渡される引数は、実行プログラム-A110aに対して、ハードウェア-A106aの有するレジスタやメモリにより受け渡されることになる。また、関数の戻り値も同様にレジスタやメモリにより受け渡される。

【0030】

そして、共通化関数インタフェース101の引数が、実行環境差異吸収プログラム-A111aに渡されるときレジスタやメモリにより受け渡される。

【0031】

以下では、それらの様子について図9を用いて説明し、そこで発生する問題点について指摘することにしよう。

【0032】

一般に、コンパイラが、CPUの汎用レジスタをデータや引数に割り付けるときには、大別してユーザに対してある関数を呼出した前後の値の存在を保証して、ユーザが使用しても良いとする（すなわち、アセンブラでレジスタの値を直接操作をしても良い）レジスタと、システム上で利用する汎用レジスタがある。この汎用レジスタは、ユーザに対しては、ある関数を呼出した前後の値の存在は保証していない。また、後者のシステム上で利用するレジスタは、ユーザの記述した引数を渡すためのレジスタも含まれる。

【0033】

汎用レジスタが、他のモジュールに値を受け渡す方法としては、汎用レジスタの中に値を直接代入するいわゆる直接形式と、メモリ上に値を格納し、汎用レジスタには、その値のアドレスを代入するいわゆる間接形式がある。また、メモリ上のエリアを一定の領域を確保し、それをスタックとし、汎用レジスタには、そのポインタを代入する方法もある。

【0034】

ここで、実行プログラム-A110aに制御が渡されるときの使用される汎用レジスタは、図9に示される様に汎用レジスタ141～149であるとする。また、それらの汎用レジスタは、以下の用途により使われるものと仮定する。

【0035】

汎用レジスタ144～146は、呼び出されたC関数の引数の格納のために使用される。これらは、システムで利用するレジスタであるため、ユーザに対しては、値の存在は保証されない。また、これらの汎用レジスタ144～146を、引数格納用途汎用レジスタ集合150と呼ぶことにする。

【0036】

汎用レジスタ142～143は、呼び出されたC関数の局所変数や一時的に使用される変数のために予約されるレジスタである。これらもシステムで利用するレジスタであるため値の保存は保証されない。

【0037】

汎用レジスタ141は、戻り値格納用途のために予約されるレジスタである。これもシステムで利用するレジスタであり、値の保存は保証されない。なお、このレジスタは、戻り値が汎用レジスタの値に収まるときには、戻り値格納用として利用され、収まらないときには、その他の局所変数や一時的変数のために使用される。

【0038】

汎用レジスタ147～148は、ユーザが利用可能なレジスタである。これらに対しては、当然、あるC関数を呼出した前後での値の存在を保証している。

【0039】

汎用レジスタ149は、スタックポインタ用途に予約されるレジスタである。このレジスタは、RAM-A151上に実装されているスタックメモリ152内をポイントするためのものである。このレジスタは、関数の呼出し前後で、値の保存は保証されている。

【0040】

さらに、コンパイラ-A108aは、RAM-A151上に、ヒープメモリ153とスタックメモリ152に分割し、それぞれのエリアを確保する。

【0041】

実行プログラム-A110aから実行環境差異吸収プログラム-A111aに制御が渡されるときに、共通化関数インターフェースに記述された引数の値が、引数格納用途汎用レジスタ集合150内の汎用レジスタ144～146のそれぞれに順に格納される。

【0042】

そして、用意している引数格納用途汎用レジスタ集合150に収まりきらなかったときには、汎用レジスタ144～146で格納できなかった共通化関数インターフェースに記述された残り全ての引数の値は、スタックメモリ152の実行プログラム-A使用スタックフレーム154上に引数格納スタック155として積まれることになる。

【0043】

さらに、共通化関数の戻り値が汎用レジスタ141のサイズよりも大きい値の場合には、実行プログラム-A使用スタックフレーム154内部に戻り値格納用途スタック156を確保し、その戻り値格納スタック156のアドレスをポイントする戻り値アドレススタック157をスタックメモリ152上に積む。

【0044】

その後、ハードウェア-A106aに搭載のPC（プログラムカウンタ）レジスタ158で指されているROM-A159上のプログラムアドレスにジャンプして、実行環境差異吸収プログラム-A111aの命令を実行する。その際、実行環境差異吸収部-A111aの命令に従って順々に、実行環境差異吸収プログラム-A使用スタックフレーム160がスタックメモリ152に格納されていく。

【0045】

最終的に、PCプログラムカウンタレジスタ158が実行環境差異吸収プログラム-A111aから実行プログラム-Aに戻る命令を指す直前には、実行環境差異吸収プログラム-A使用スタックフレーム160は、スタックメモリ152からは解放されることになる。

【0046】

なお、実行プログラム-A使用スタックフレーム154と実行環境差異吸収部-A使用スタックフレーム160で使われているスタックフレームとは、呼び出されたC関数の局所変数の値を一時的に格納するためのスタックメモリの集合体であり、コンパイラが予めスタックメモリ上に予約する領域である。

【0047】

ところで、このような従来技術については、共通化関数を呼出すときの共通化関数インタフェースの引数の数と、実行環境差異吸収プログラム-Aに制御を渡すときの引数格納用途汎用レジスタ集合150内の汎用レジスタ144~146の数との関係については考慮されていない。

【0048】

すなわち、共通化関数インターフェース101の引数の数が、引数格納用途汎用レジスタ集合150内の汎用レジスタの数を下回る場合には良いが、引数格納用途汎用レジスタ集合150内の汎用レジスタの数を上回る場合には、コンパイラによって格納できない引数のデータを、スタックメモリ152内の引数格納スタック155に積むようなコードが展開される。

【0049】

そのため、実行プログラム-A110aから実行環境差異吸収プログラム-A111aに制御を渡すときには、スタックメモリ152内の引数格納スタック155に積む処理が余分に入り、実行速度が遅くなるという問題点があった。

【0050】

スタックメモリ152の使用効率の観点からしても、汎用レジスタ144~146に格納できなかった引数の数だけ、スタックメモリ152に格納し、積み上げる分、RAM151の容量が増加し、RAMの使用効率も悪くなるという問題点もあった。

【0051】

本発明は、上記問題点を解決するためになされたもので、その目的は、異なった実行環境で実行させるプログラムを開発するときに、実行環境差異吸収プログラムを呼出すときの汎用レジスタの割当てを考慮することにより、メモリの使用効率が良く、しかも、実行速度が低下しない異種実行環境におけるレジスタの割

当て方法、異種実行環境におけるソフトウェア開発方法、および、それを実行するプログラムが組み込まれたLSIを提供することにある。

【0052】

【課題を解決するための手段】

本発明では、共通化関数の引数の数を減らし、実行環境差異吸収プログラムAに制御が渡るときに、メモリ上に引数情報集合部を作って、これを汎用レジスタにポイントさせるようにする。

【0053】

このようにすれば、実行時にスタックメモリ152の引数格納スタック155にデータを積む処理が省略できるため、実行時の関数呼出しによるオーバーヘッドが軽減される。

【0054】

また、従来技術で引数格納スタック155を利用する場合には、スタックの使用量が増加する恐れがあったが、本発明では、RAM上に連続した引数情報集合部を作るので、メモリの使用効率を良くすることもできる。

【0055】

このためには、共通化関数の引数の数は、コンパイラが引数格納用として予約する引数格納用途汎用レジスタ集合150の数より越えないようにしておく必要がある。

【0056】

【発明の実施の形態】

〔開発環境とハードウェア〕

先ず、図1および図2を用いて本発明に係る異種実行環境におけるレジスタの割当て方法のソフトウェア開発環境と、それを動作させるためのハードウェアについて説明する。なお、本実施形態では、ソースプログラムの記述を、C言語でおこなった場合を想定するものとする。

【0057】

図1は、本発明に係る異種実行環境におけるレジスタの割当て方法のソフトウェア開発環境の処理の流れを説明するための図である。

【0058】

図2は、本発明に係る異種実行環境におけるレジスタの割当て方法を実行するためのハードウェア構成図である。

【0059】

本発明は、複数の異なったOSでの開発環境180を前提としている。

【0060】

ソフトウェアの開発者は、アプリケーションのプログラムが記述したアプリケーション本体100と、共通化関数のソースコードを入力して、通常の開発手順の通り、プリプロセッサ181、Cコンパイラ182、アセンブラ183、リンカ184と言うソフトウェアツールにより、実行プログラムを作成する。

【0061】

プリプロセッサ181は、条件によって加工したり、読み飛ばしたり、他プログラムを組み込んだりマクロ定義処理などをおこなうCコンパイラに入る前の前処理のためのツールである。Cコンパイラ182は、記述されたテキストコードに対して、構文解析・意味解析などをおこなった、適当なアセンブラコードを生成するものである。アセンブラ183は、アセンブラコードをCPU200で実行させる機械語に変換するツールである。リンカ184は、ライブラリ190から適当なプログラムを選び、数種のプログラムファイルを結合して最終的にハードウェア上で実行させる実行プログラムを作成する。このライブラリ190には、実効環境に依存する実行環境プログラムが含まれていて、機能に応じてリンカ184により適当なプログラムが取りこまれて結合される。

【0062】

OSコンフィグレータ191は、アプリケーションソフトウェア開発者に、ターゲットとするOS107の諸機能を選択および設定させカスタマイズ化されたOS107のプログラムコードを作成するソフトウェアツールである。

【0063】

ハードウェア106は、CPU200、Device201、ワーキングメモリ202から構成されている。

【0064】

最終的にCPU200上で実行する実行プログラムとOS107は、ワーキングメモリ202に配置される。

【0065】

本実施形態のソフトウェア開発環境は、メインフレーム、パーソナルコンピュータやワークステーション等の汎用計算機上に実装することが可能である。

【0066】

最終的に実行プログラムを動作させることのできるハードウェア106を作成する手順を示すと以下の様になる。

【0067】

すなわち、アプリケーションソフトウェア開発者は、汎用計算機上に定義されているターゲットのOS107に対応するOS別プログラム開発環境180を選択する。そして、選択されたOS別プログラム開発環境180に付随しているOSコンフィグレータ191を用いてカスタマイズしたOS107のプログラムコードを作成する。

【0068】

次に、そのOS107のプログラムコードと実行プログラム110と実行環境差異吸収プログラム111をリンクしてできあがったプログラムコード全体をワーキングメモリ202に組み込む。また、予めOS107のプログラムコードが搭載されたワーキングメモリ202に、後から実行プログラム110と実行環境差異吸収プログラム111をリンクしてできあがったプログラムコードを組み込むようにしても良い。

【0069】

ここで、ハードウェア106は、実行プログラムを実行できるような機構を備えていれば良いが、本実施形態では、主にプログラムをワーキングメモリに組み込んだシステムLSIなどを想定している。

【0070】

以下、図2を用いてこのハードウェア106の構成の例について詳細に説明する。

【0071】

CPU200は、命令を実行し、計算をおこなうハードウェア106のメインのコンポーネントであり、図9で説明したような汎用レジスタ141～149を備えている。

【0072】

RAM202aは、いつでも書き込みが可能なメモリエリアである。ROM202bは、通常は書きこむことができず、読みだし専用のメモリエリアである。例えば、LSIの製造時にのみ情報を書きこみ、通常の実行時には、このメモリエリアからは、情報を読むだけである。上記の実行プログラムとOS107は、RAM202aかROM202bに配置され、CPU200に読み出されて実行される。

【0073】

デバイス201は、各種機能を持つハードウェア106のコンポーネントであり、ハードウェア106がシステムLSIの場合には、LSI上に構成される特殊機能の処理回路である。具体的には、I/O (Input/Output) , ASIC (Application Specific Integrated Circuits) , FPGA (Field Programmable Gate Arrays) , DSP (Digital Signal Processor) などが考えられる。

【0074】

I/Oは、例えば、A/D変換器、D/A変換器、RS232-C処理回路、SCSI処理回路である。ASICは、例えば、MPEG Video符号器、MP3復号器等の専用処理回路である。FPGAは、ハードウェア構成を可変にできるICを言う。DSPは、デジタル信号処理専用のICである。

【0075】

これらのコンポーネントは、信号の共通の通り道であるバス203により、情報のやり取りをする。

〔汎用レジスタの割当て方法〕

次に、従来技術の説明の所で述べたことを前提として、図3を用いて本発明に係る汎用レジスタの割当て方法について説明する。

【0076】

図3は、本発明に係る汎用レジスタの割当て方法による実行環境差異吸収プロ

グラム-Aを呼出すときのレジスタの割当てを示す図である。

【0077】

従来技術では、図9で示したように実行環境差異吸収プログラム111に制御が渡される際の渡すべき引数、具体的には、共通化関数の引数の数が、引数格納用途汎用レジスタ集合150の数を越えるときに、スタックメモリ152内の引数格納スタック155に積まれることで問題が生じることを指摘した。

【0078】

そこで、本発明では、図3に示されるように、実行環境差異吸収プログラム-A111aを呼出すときに、RAM151上に引数情報集合部161を作り、そこをポイントするようにして、引数格納スタック155には、データを積まない様に工夫する。このようにすれば、引数情報集合部161には、実行環境差異吸収プログラム-A111aを呼出した時点で、引数の値が設定されているため、実行プログラム-A110aがおこなうスタックメモリ152への引数格納スタック155の積み上げとそれを解放する作業をおこなわなくても良くなり、実行時のオーバヘッドを低減することができる。また、連続した領域のデータを間接的にポイントするようなインタフェースにしているためRAMの使用効率も向上する。

〔汎用レジスタの割当て方法のコーディングをふまえた具体例〕

次に、図4および図5を用いて本発明の汎用レジスタの割当て方法を、C言語のコーディングをした場合の具体例を、より詳細に説明する。

【0079】

図4は、本発明の汎用レジスタの割当て方法を用いる場合のC言語によるコーディングの具体例を示す図である。

【0080】

図5は、図4のコーディングをした場合の実行環境差異吸収プログラム-Aを呼出すときのレジスタの割当てを示す図である。

【0081】

本実施形態で説明するC言語によるコーディングの共通化関数102の例は、WRP_Task_Create229である。この関数は、タスクを生成する関数であるとす

る。そして、この関数の中で、タスクルーチン関数Routine 2 2 3 が呼ばれているものとする。この関数には、タスク処理そのものが記述される。

【 0 0 8 2 】

共通化関数WRP_Task_Create 2 2 9 の引数は、4 つである。また、戻り値は、Err 2 3 7 という変数用のRAM 1 5 1 に格納される。

【 0 0 8 3 】

第一の引数 2 5 0 は、TSKID型のtid 2 3 1 のアドレスであり、タスク識別子のポインタが渡される。タスク識別子は、OS 1 0 7 が生成するタスクを識別するために用いる。tid 2 3 1 には、OS 1 0 7 がタスクを生成した際に生成されるタスク識別子が格納される。

【 0 0 8 4 】

第二の引数 2 5 1 は、void型の関数のRoutine 2 3 3 というタスクルーチンのアドレスであり、タスクルーチンポインタが渡される。なお、引数に関数の関数名を書くことにより、その関数のアドレスを渡すと言うのがC言語の言語仕様である。

【 0 0 8 5 】

第三の引数 2 5 2 は、int型のPriority 2 3 2 であり、タスク優先度が渡される。

【 0 0 8 6 】

第四の引数 2 5 3 は、ENV_INFO型のデータenvironment 2 3 0 のアドレスであり、共通化関数に渡すための環境情報へのポインタである。

【 0 0 8 7 】

以下、この環境情報について詳細に説明する。

【 0 0 8 8 】

図 4 に示したCコーディング 3 0 0 では、typedef文で、ENV_INFO型構造体に定義されている。

【 0 0 8 9 】

TSK_ATTRB型のAttribution 2 2 2 は、タスク属性値である。タスク属性値は、アプリケーション本体 1 0 0 がC言語で記述されているのか、アセンブラ言語で

記述されているのか、および、浮動小数点演算プロセッサ、デジタル信号処理プロセッサを使用するのか、しないのか等の設定する値である。

【0090】

int型のQuantum223は、プロセッサ上限時間値である。プロセッサ上限時間値は、タスクがCPU200を占有する上限の時間幅を設定する値である。

【0091】

int型のStackSize224は、スタックサイズである。スタックサイズは、タスクを実行させる際に最低限必要となるスタックメモリ152のサイズである。

【0092】

void*型のStackBasePtr225は、スタックベースポインタである。スタックベースポインタは、タスクの初期化時のスタックアドレス値である。

【0093】

int型のCPU_Mode226は、CPUモード値である。CPUモード値は、生成されるタスクにおいて、CPU200がマルチメディア命令を使用するかしないかを設定する値である。

【0094】

char*型のName227は、タスク名称格納先アドレスである。タスク名称格納先アドレスは、生成するタスクの名称となる文字列の格納先のアドレスの値である。

【0095】

void*型のTo_Task228は、タスク情報アドレスである。タスク情報アドレスは、OS107がタスクを生成するときに渡される情報のアドレスの値である。

【0096】

図のCコーディング300では、ENV_INFO型データとして、environment230という変数を定義している。特に、OS107間でアプリケーション本体100を再利用したり、移植したりする際には、OS別プログラム開発環境180やハードウェア106毎に、ENV_INFO型構造体221内の定義情報を変更すれば、容易に異なるOS107やハードウェア106上でアプリケーション本体100を実装できる。

【0097】

また、タスクルーチンであるRoutine233には、To_Task228と言う引数を渡すことを示している。

【0098】

このソースコードをコンパイルして、実行プログラムを作成し、それを適当なOS107で実行させると、実行環境差異吸収プログラムに、共通化関数239で記述した4つのデータが渡される。すなわち、タスク識別子へのポインタ、タスクルーチンへのポインタ、タスク優先度、環境情報へのポインタである。

【0099】

実行環境差異吸収プログラム-A111aの中で、OS107aのシステムコールを呼び出してOSやハードウェアの機能を利用することになる。この例では、生成するタスクルーチンのポインタを渡し、タスクを生成するシステムコールを呼出すことになる。

【0100】

次に、図5により共通化関数を呼び出したときの実行プログラムがハードウェア資源を占有している状態を説明する。

【0101】

ここで、CPU200の引数格納用途汎用レジスタ集合150内の汎用レジスタの数が予め4つであるとする。

【0102】

このときには、共通化関数WPR_Task_Create229の引数は、図5の様にレジスタ350～353に順に格納される。すなわち、上から順に、タスク識別子ポインタ350、タスクルーチンポインタ351、タスク優先度値352である。

【0103】

環境情報ポインタ353は、RAM上の環境情報をポイントしていて、ENV_INF0型の構造体で定義した情報が、RAM上の450～456に上ら順に格納される。すなわち、上から順に、タスク属性値450、プロセッサ上限時間値451、スタックサイズ452、スタックベースポインタ453、CPUモード値454、タスク名称格納先ポインタ455、タスク情報格納先ポインタ456である。

【0104】

このエリアは、図3で説明したところの引数情報集合部に該当する。

【0105】

このように、引数情報集合部のエリアを作ることにより、引数格納用途汎用レジスタ集合150に引数を取めることができるため、引数格納スタック155を使用することがない。

【0106】

【発明の効果】

本発明によれば、異なった実行環境で実行させるプログラムを開発するときに、実行環境差異吸収プログラムを呼出すときの汎用レジスタの割当てを考慮することにより、メモリの使用効率が良く、しかも、実行速度が低下しない異種実行環境におけるレジスタの割当て方法、異種実行環境におけるソフトウェア開発方法、および、それを実行するプログラムが組み込まれたLSIを提供することができる。

【図面の簡単な説明】

【図1】

本発明に係る異種実行環境におけるレジスタの割当て方法のソフトウェア開発環境の処理の流れを説明するための図である。

【図2】

本発明に係る異種実行環境におけるレジスタの割当て方法を実行するためのハードウェア構成図である。

【図3】

本発明に係る汎用レジスタの割当て方法による実行環境差異吸収プログラムAを呼出すときのレジスタの割当てを示す図である。

【図4】

本発明の汎用レジスタの割当て方法を用いる場合のC言語によるコーディングの具体例を示す図である。

【図5】

図4のコーディングをした場合の実行環境差異吸収プログラム-Aを呼出するときのレジスタの割当てを示す図である。

【図6】

特開平8-63363号公報の「仮想実行環境システム」のシステム概要図である。

【図7】

異なった実行環境でソフトウェア開発をおこなうときの概念図である。

【図8】

計算機システム上で実行プログラムが実行されるときに呼出し関係を示した図である。

【図9】

実行環境差異吸収プログラム-Aを呼出するときのレジスタの割当てを示す図である。

【符号の説明】

- 10…アプリケーションソフトウェア
- 11…プログラム本体
- 12…置換情報記述部
- 20…仮想実行環境実現システム
- 21…翻訳部既存OS用コンパイラ
- 22…実行部
- 30…実行可能プログラム
- 50…既存CPU
- 60…既存OS
- 100…アプリケーション本体
- 101…共通化関数
- 102…共通化インターフェース
- 103…実行環境プログラム
- 104…実行環境
- 105…計算機システム

- 106…ハードウェア
- 107…OS
- 108…コンパイラ
- 109…リンカ
- 110…実行プログラム
- 111…実行環境差異吸収プログラム
- 121…システムコール
- 141～149…汎用レジスタ
- 150…引数格納用途汎用レジスタ集合
- 151…RAM
- 152…スタックメモリ
- 153…ヒープメモリ
- 154…実行プログラム使用スタックフレーム
- 155…引数格納スタック
- 156…戻り値格納スタック
- 157…戻り値アドレススタック
- 158…PCレジスタ
- 159…ROM
- 160…実行環境差異吸収プログラムスタックフレーム
- 161…引数情報集合部
- 180…OS別プログラム開発環境
- 181…プリプロセッサ
- 182…Cコンパイラ
- 183…アセンブラ
- 184…リンカ
- 190…ライブラリ
- 191…OSコンフィグレータ
- 200…CPU
- 201…デバイス

202…ワーキングメモリ
 202a…RAM
 202b…ROM
 203…バス
 221…ENV_INFO型構造体
 222…Attribution (タスク属性値)
 223…Quantum (プロセッサ上限時間値)
 224…StackSize (スタックサイズ)
 225…StackBasePtr (スタックベースポインタ)
 226…CPU_Mode (CPUモード値)
 227…Name (タスク名称格納先アドレス)
 228…To_Task (タスク情報アドレス)
 229…WR_Task_Create (タスク作成用途共通化関数)
 230…environment (ENV_INFO型データ)
 231…tid (タスク識別子)
 232…priority (タスク優先度)
 233…Routine (タスクルーチン)
 235…タスクルーチンポインタ
 237…変数
 250～253…共通化関数に渡す第一ないし第四引数
 300…Cコーディング
 350…タスク識別子ポインタ
 351…タスクルーチンポインタ
 352…タスク優先度値
 353…環境情報ポインタ
 450…タスク属性値
 451…プロセッサ上限時間値
 452…スタックサイズ
 453…スタックベースポインタ

454…CPUモード値

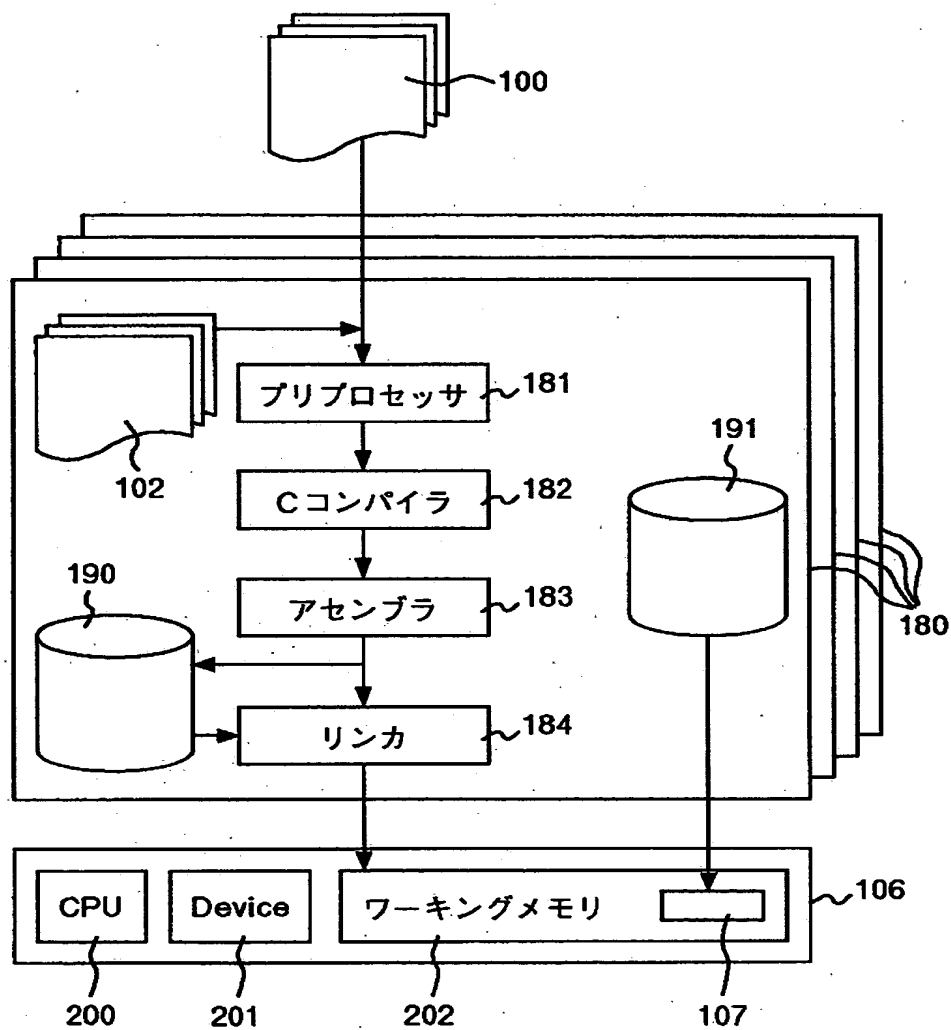
455…タスク名称格納先ポインタ

456…タスク情報格納先ポインタ

【書類名】 図面

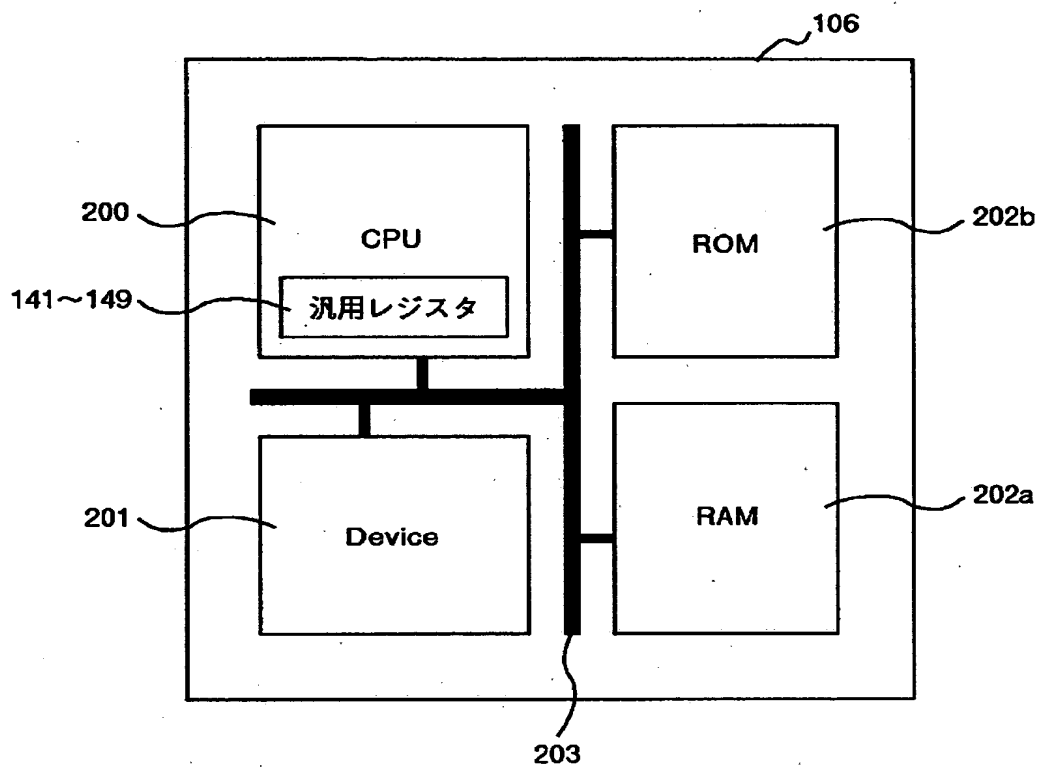
【図 1】

図 1



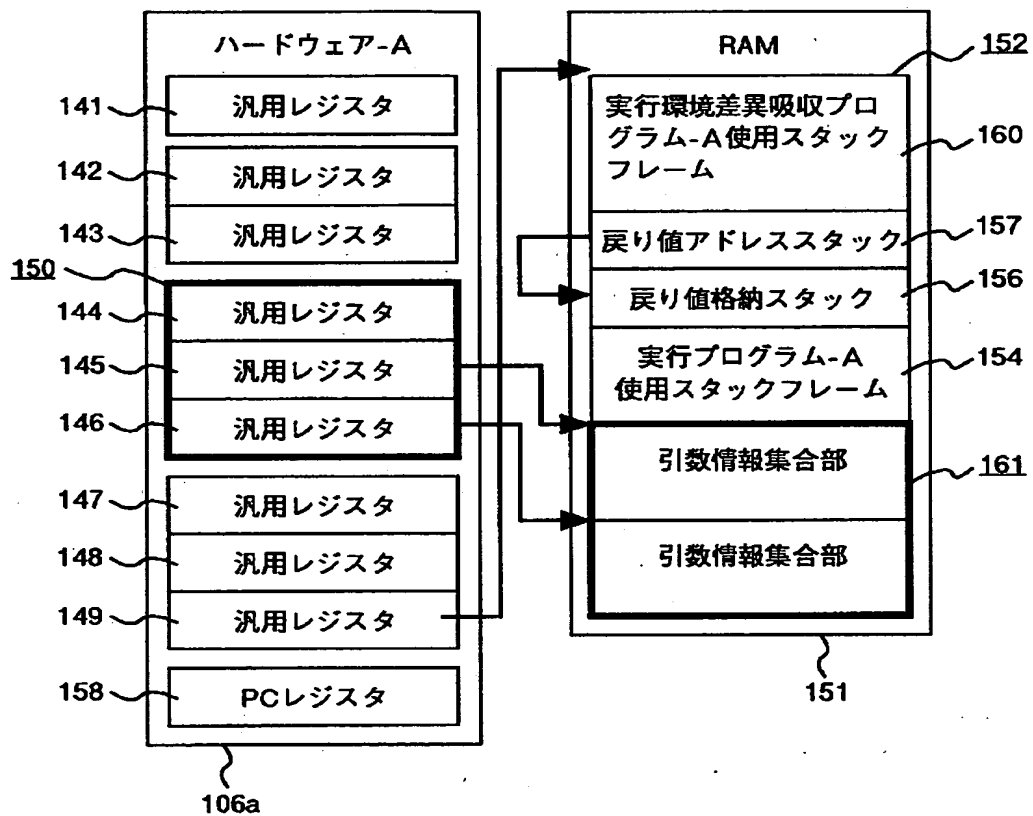
【図2】

図 2



【図 3】

図 3



【図 4】

図 4

300

```

typedef      struct {
    TSK_ATRB      Attribution; 222
    int           Quantum;     223
    int           StackSize;   224
    void *        StackBasePtr; 225
    int           CPU_Mode;    226
    char *        Name;        227
    void *        To_Task;     228
} ENV_INFO;

ENV_INFO      environment; 230
TSKID         tid;          231
int           priority;     232

void Routine ( void * To_Task ) 233
{
    .....
}

Er = WRP_Task_Create ( &tid, Routine, priority, &environment ); 229
    
```

221 {

237 {

250 {

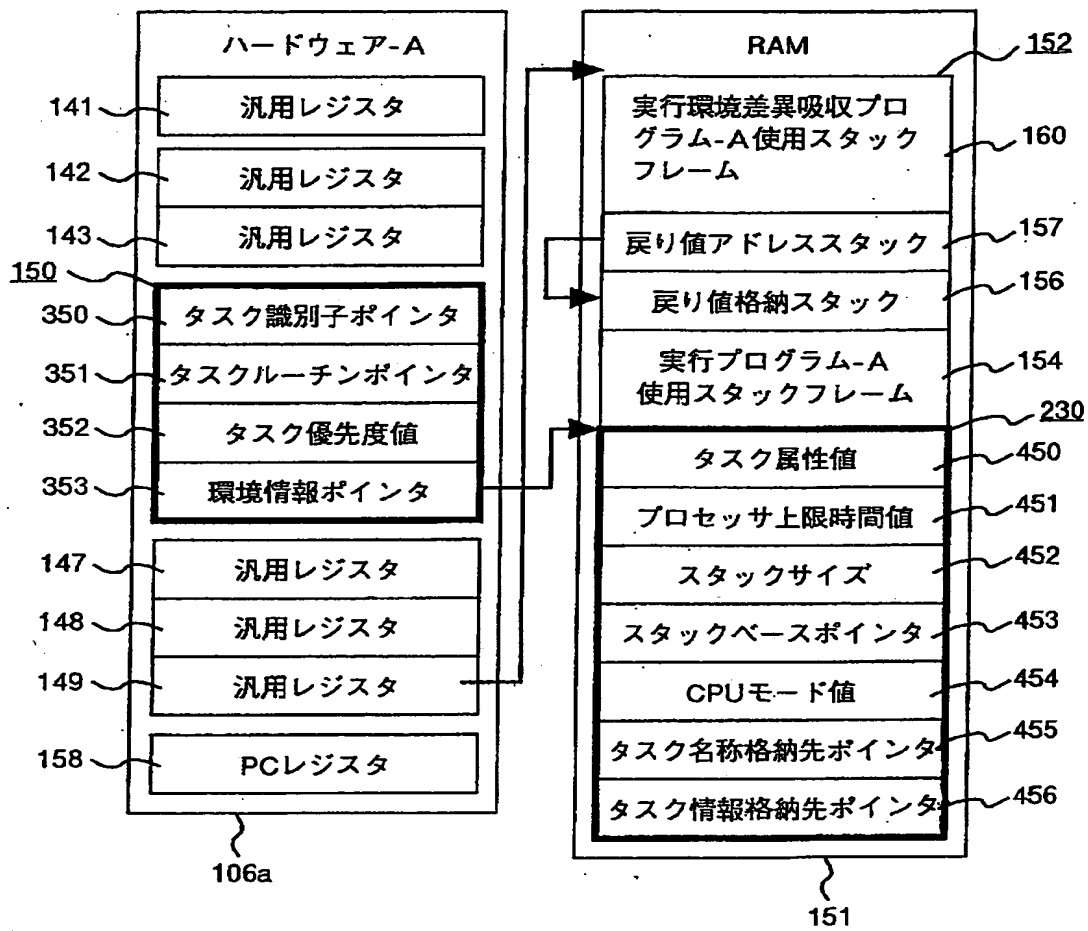
251 {

252 {

253 {

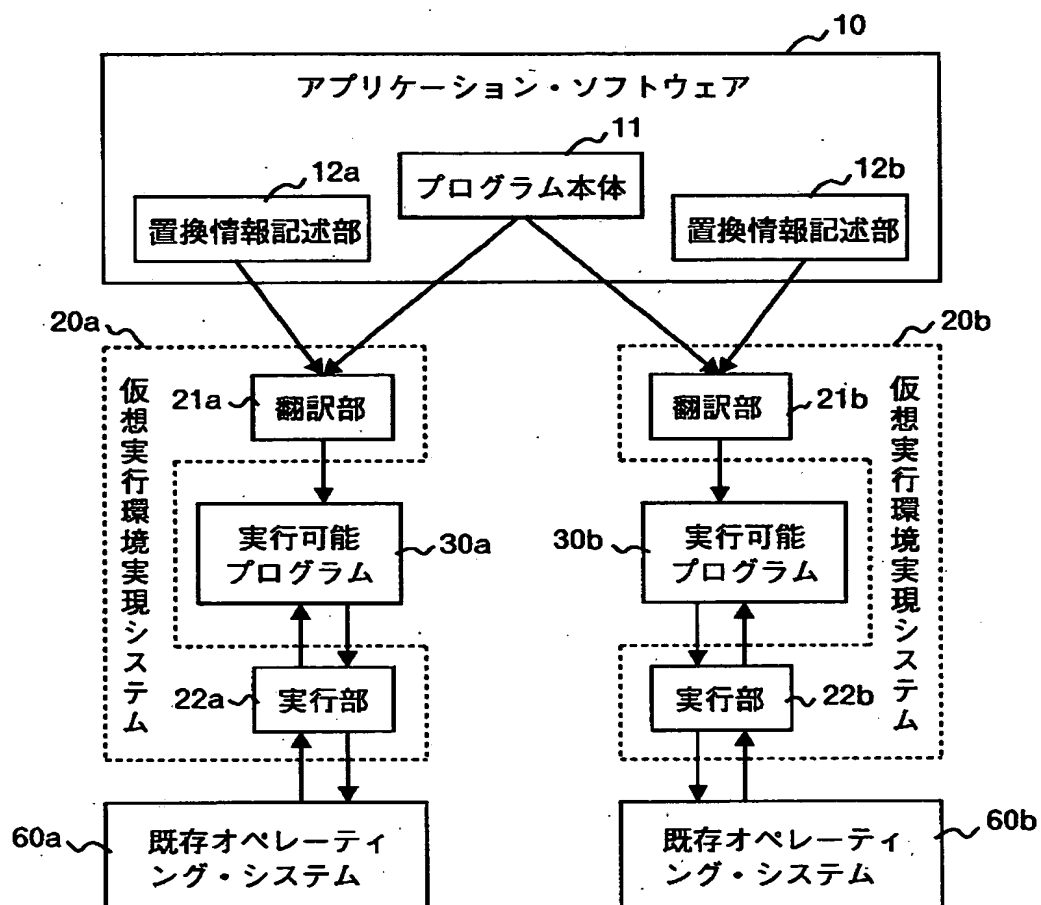
【図5】

図 5



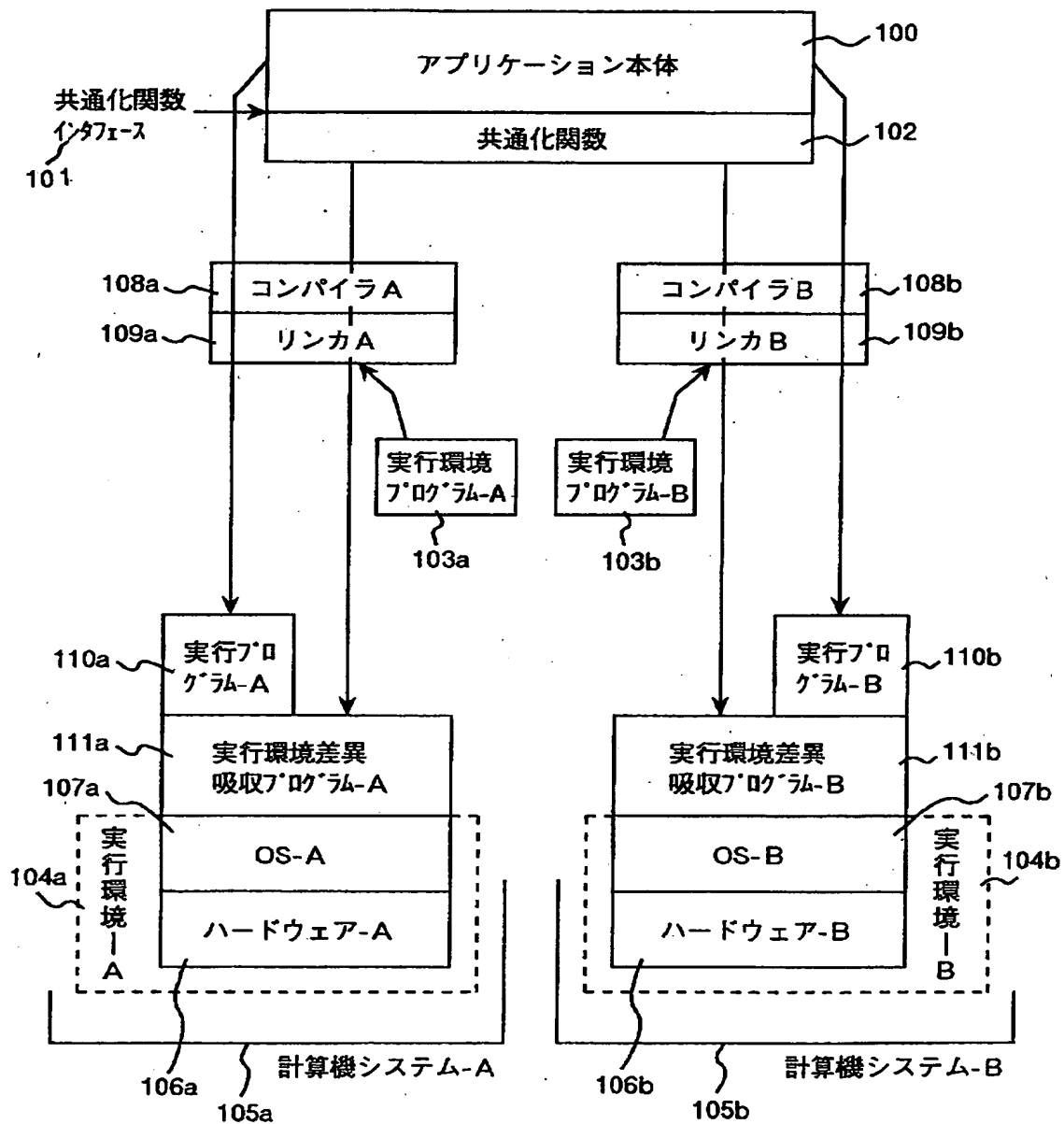
【図 6】

図 6



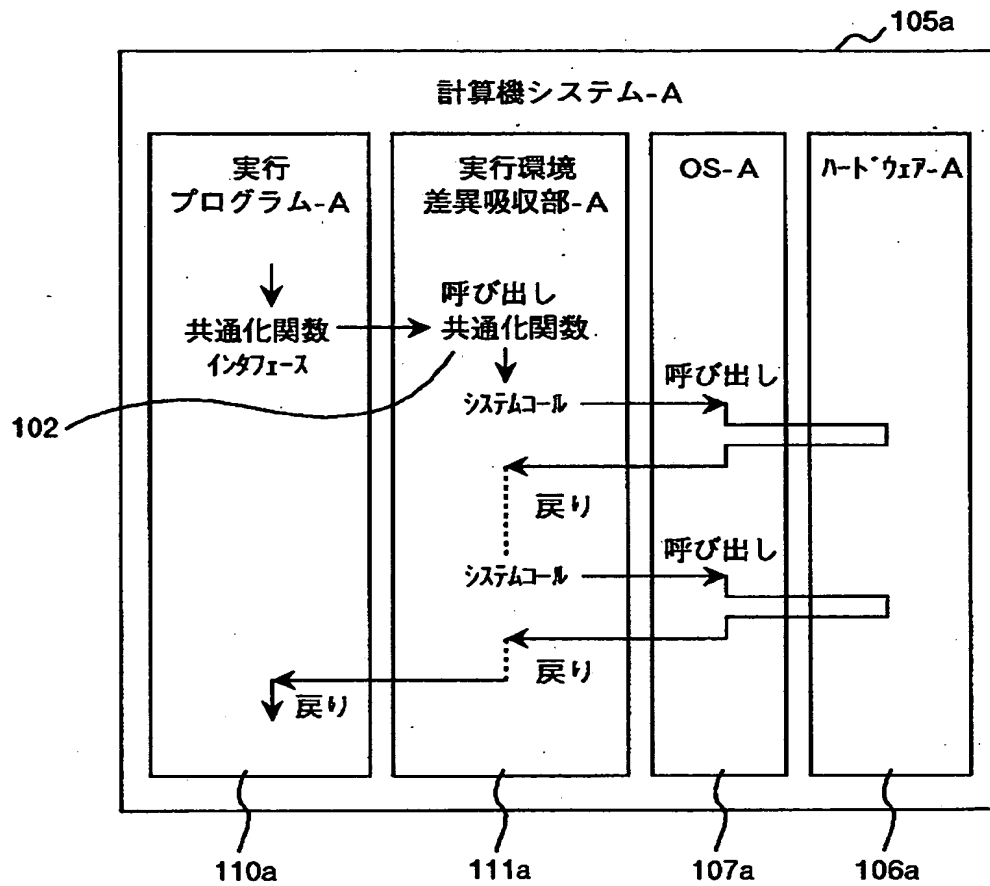
【図 7】

図 7



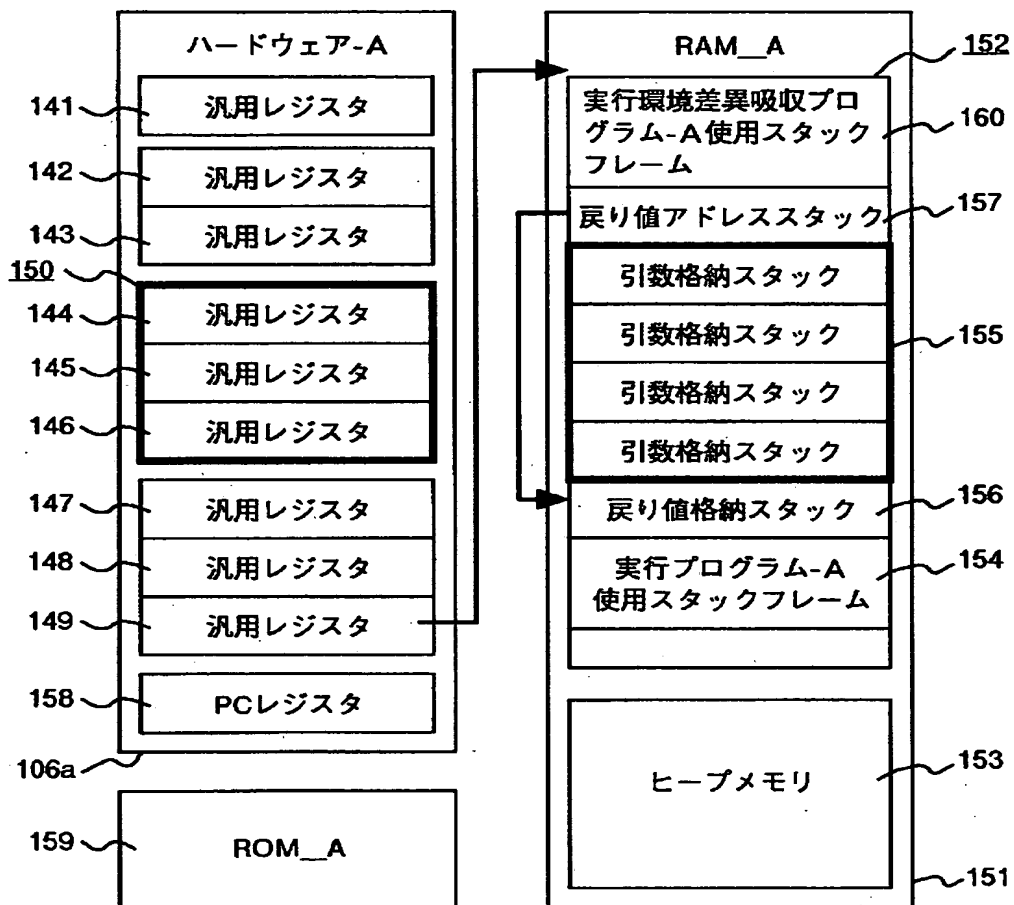
【図 8】

図 8



【図 9】

図 9



【書類名】 要約書

【要約】

【課題】 異なった実行環境で実行させるプログラムを開発するときに、実行環境差異吸収プログラムを呼出すときの汎用レジスタの割当てを考慮することにより、メモリの使用効率を良くし、しかも、実行速度が低下しないようにする。

【解決手段】 異種実行環境におけるレジスタの割当て方法において、実行環境差異吸収プログラムに制御を渡すときのパラメータの数を、メモリに引数格納集合部を作り、汎用レジスタにその引数格納集合部をポイントさせることにより、コンパイラが予約する引数用途格納用レジスタの数を越えない様に調整する。また、共通化関数の引数の数を、コンパイラが予約する引数用途格納用レジスタの数を越えない様にする。

【選択図】 図 3

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都千代田区神田駿河台4丁目6番地
氏 名 株式会社日立製作所